**ASSIGNMENT-5**

**MODULE-4 Automation core testing (Load Runner Up and selenium IDE)**

**Q-1 Which components have you used in Load Runner?**

**ANS..** Load Runner is a performance testing tool developed by Micro Focus. It consists of several components that work together to conduct performance testing on various software applications. In Load Runner's automation core testing, the main components typically used are:

Vu Gen (Virtual User Generator): Vu Gen is used to create scripts that simulate user interactions with the application under test. These scripts are recorded or manually coded to mimic real user behavior.

Controller: The Controller component is used to manage and control the load test scenarios. It allows you to define the behavior of virtual users, set up scenarios, distribute the load across multiple machines, and monitor the test execution.

Load Generators: Load Generators are machines or virtual machines where virtual users are executed. These machines generate the load on the application by simulating multiple users accessing the application simultaneously.

Analysis: The Analysis component is used to analyze the performance test results. It provides various graphs, reports, and statistics to evaluate the performance of the application under different load conditions.

ALM (Application Lifecycle Management): ALM integration allows for the management of test assets, such as scripts, test scenarios, and results, within a central repository. It facilitates collaboration among team members and provides traceability of testing activities.

Performance Center: Performance Center is an enterprise-level solution for managing and executing performance tests. It provides centralized management of test resources, scheduling of tests, and comprehensive reporting capabilities.

**Q-2 How can you set the number of Vusers in load runner?**

**ANS..** In Load Runner, Vusers (Virtual Users) are simulated users that emulate real users interacting with your application under load. The number of Vusers you set depends on the load you want to simulate on your system during performance testing.

To set the number of Vusers in Load Runner, you typically follow these steps:

1. Open Load Runner Controller: Load Runner provides a Controller component that allows you to manage and execute your performance tests.

2. Create a Scenario: In Load Runner, you create a scenario which defines the characteristics of the load test you want to conduct. This includes details such as the number of Vusers, script details, test duration, and more.

3. Define Vuser Groups: Within your scenario, you define Vuser groups. A Vuser group is a collection of Vusers that execute the same script. Each Vuser group can have its own settings such as the number of Vusers, iteration count, and pacing.

4. Set the Number of Vusers: Within each Vuser group, you specify the number of Vusers you want to simulate. This can be a fixed number or can be set to vary dynamically over time to simulate different levels of load.

5. Configure Ramp-up: In many cases, you may not want to start all Vusers simultaneously. Instead, you might want to ramp up the load gradually over time. Load Runner allows you to configure ramp-up settings within your scenario.

6. Execute the Test: Once you have configured your scenario and set the number of Vusers, you can execute your performance test using the Controller.

7. Monitor and Analyze Results: During test execution, you can monitor various performance metrics such as response times, throughput, and resource utilization. After the test completes, you can analyze the results to identify any performance bottlenecks or issues.

**Q-3 What is Correlation?**

**ANS..** In the context of automation testing, correlation refers to the process of identifying and managing dynamic values within a web application that may change from one session to another or during runtime. These dynamic values are often generated by the server and are crucial for maintaining session integrity, security, and proper functionality of the application.

Correlation is important in automation testing because automated scripts need to be able to handle these dynamic values in order to accurately simulate real user interactions with the application. Without proper correlation, automated tests may fail due to mismatches between expected and actual values.

Common examples of dynamic values that require correlation in automation testing include session IDs, authentication tokens, timestamps, and unique identifiers generated by the server.

The process of correlation typically involves capturing the dynamic values from server responses during test execution and then substituting them with placeholders or variables in subsequent requests. This allows the automated test script to maintain the session state and continue executing the test scenario accurately.

Various automation testing tools provide built-in mechanisms or plugins to handle correlation automatically, simplifying the process for testers. However, understanding the concept of correlation and how to manage dynamic values manually is still important for more complex testing scenarios or when using tools that may not offer built-in support for correlation.

**Q-4 What is the process for developing a Vuser Script?**

**ANS..** Developing a Vuser script, commonly used in performance testing with tools like LoadRunner, JMeter, or other similar performance testing tools, involves several steps. Here's a typical process:

Understanding Requirements: Begin by understanding the requirements of the performance test. You need to know what actions the Vuser (virtual user) should perform during the test, what workflows it should follow, and what load it should simulate.

Recording: Typically, you start by recording the actions of a real user using the application. During this recording phase, the tool captures the interactions between the user and the application under test. These interactions include HTTP requests, form submissions, clicks, etc.

Enhancing the Script: After recording, you'll need to enhance the script to make it robust and realistic. This might involve parameterization (replacing hard-coded values with variables), correlation (capturing dynamic data from responses and passing them to subsequent requests), and adding logic (e.g., branching, loops) to mimic real user behavior.

Parameterization: Parameterization is essential for creating realistic load scenarios. It involves replacing fixed values in the script with parameters that can be dynamically generated or retrieved from data sources such as CSV files, databases, or other external sources.

Correlation: Many web applications use dynamic data, such as session IDs or timestamps, which need to be correlated during script execution

to ensure accurate simulation of user behavior. Correlation involves extracting dynamic data from server responses and passing it as inputs to subsequent requests.

Adding Think Time: Think time represents the time a real user spends between interactions with the application, such as reading content or considering options before clicking. Adding realistic think time to the script helps simulate actual user behavior and produces more accurate test results.

Parameterizing Load: Once the script is ready, you'll need to configure the load profile for the performance test. This includes specifying the number of virtual users, ramp-up and ramp-down periods, and other relevant parameters to simulate the desired load on the application.

Script Debugging and Validation: Before executing the test, it's essential to debug and validate the script to ensure it behaves as expected. This may involve running the script in debug mode, checking for errors, and validating that the script accurately simulates the intended user interactions.

Executing the Test: Once the script is validated, you can execute the performance test. During the test execution, the performance testing tool will simulate the specified load on the application by running multiple instances of the Vuser script concurrently.

Analyzing Results: After the test execution completes, you'll need to analyze the results to identify any performance issues such as response time degradation, errors, throughput limitations, etc. This analysis helps in identifying bottlenecks and optimizing the application for better performance.

Iterative Optimization: Based on the analysis results, you may need to iterate on the script, load profile, or application configuration to optimize performance further. This iterative process continues until the desired performance goals are achieved.

**Q-5 How load Runner interacts with the application?**

**ANS..** Load Runner is a performance testing tool that simulates user activity on an application to measure its performance under various load conditions. It interacts with the application under test in the following ways:

Scripting: Load Runner allows testers to create scripts that simulate user actions such as clicks, form submissions, data inputs, etc. These scripts are typically recorded using protocols such as HTTP, HTTPS, Web Services, etc., depending on the type of application being tested.

Parameterization: Load Runner provides features to parameterize the scripts, allowing testers to simulate multiple users with different data sets. This helps in mimicking real-world scenarios where users interact with the application using different inputs.

Correlation: When recording scripts, Load Runner captures dynamic data such as session IDs, tokens, etc. This data needs to be correlated to ensure that subsequent virtual users do not use the same data, thus ensuring realistic test scenarios.

Test Execution: Once the scripts are prepared, Load Runner orchestrates the execution of these scripts on multiple virtual users. It controls the timing and sequencing of the requests to simulate realistic user behavior.

Monitoring: During test execution, Load Runner monitors various performance metrics such as response times, throughput, server resource utilization, etc. It collects data from the application, servers, and network infrastructure to analyze the performance under load.

Analysis: After the test execution, Load Runner provides detailed reports and analysis tools to identify performance bottlenecks, resource constraints, and other issues. Testers can analyze the data to optimize the application's performance and scalability

**Q-6** **How many VUsers are required for load testing?**

**ANS..** The number of virtual users (VUsers) required for load testing depends on various factors such as the application's architecture, expected user traffic, system resources, and performance goals. There isn't a one-size-fits-all answer to this question. However, here are some general steps to determine the number of VUsers required:

Understand the Application: Gain a deep understanding of the application being tested, including its architecture, components, and expected usage patterns.

Define Performance Goals: Determine the performance metrics that need to be achieved, such as response time, throughput, and resource utilization thresholds.

Identify Scenarios: Identify the critical user scenarios or workflows that need to be tested under load. These scenarios should represent real-world usage patterns.

Determine Concurrent Users: Estimate the number of concurrent users expected to access the application during peak usage periods. This can be based on historical data, user surveys, or business projections.

Calculate VUsers: Based on the identified scenarios and concurrent user estimates, calculate the number of VUsers needed to simulate the expected load.

Start Conservatively: Start with a conservative estimate and gradually increase the number of VUsers until performance goals are met or until system limitations are reached.

Monitor Performance: Continuously monitor system performance during load testing to identify bottlenecks, scalability issues, or performance degradation.

Iterate and Optimize: Iterate on the load testing process, adjusting the number of VUsers and refining scenarios based on test results. Optimize the application and infrastructure to improve performance where necessary.

**Q-7 What is the relationship between response time and throughput?**

**ANS..** Response Time: Response time, also known as latency, is the time taken for a system to respond to a user request. It measures the time elapsed between when a user action is initiated and when the system provides a response. In the context of automation core testing, response time indicates how quickly the system under test can process requests and return results.

Throughput: Throughput refers to the rate at which a system can process a certain number of requests or transactions within a given period. It is typically measured in terms of requests per second (RPS) or transactions per second (TPS). Throughput measures the system's capacity to handle a workload efficiently.

The relationship between response time and throughput can be understood as follows:

Inverse Relationship: In general, there is an inverse relationship between response time and throughput. As response time decreases (i.e., the system responds faster to requests), throughput tends to increase (i.e., the system can process more requests within a given time frame). Conversely, as response time increases, throughput typically decreases because the system is slower in processing requests.

Optimization Balance: Achieving optimal performance involves striking a balance between response time and throughput. While it's important to minimize response time to ensure users receive prompt responses, excessively optimizing response time may negatively impact throughput by overloading the system. Conversely, prioritizing throughput at the expense of response time may lead to degraded user experience due to increased latency.

Performance Tuning: During automation core testing, performance engineers often focus on optimizing both response time and

throughput. This may involve identifying bottlenecks, optimizing code, fine-tuning hardware resources, and implementing caching or load balancing strategies to improve system performance.

**Q-8 To test the performance testing on “Tops Technologies website”:- <http://www.saucedemo.com/>**

1. **To record all top level menu**
2. **To record minimum 10 Vuser on this website**
3. **Save all (Script, Design, Graph)**

**ANS..**  Action()

{

web\_url("gts1c3.der",

"URL=http://pki.goog/repo/certs/gts1c3.der",

"Resource=1",

"RecContentType=application/pkix-cert",

"Referer=",

"Snapshot=t5.inf",

LAST);

web\_url("gtsr1.der",

"URL=http://pki.goog/repo/certs/gtsr1.der",

"Resource=1",

"RecContentType=application/pkix-cert",

"Referer=",

"Snapshot=t6.inf",

LAST);

web\_url("gts1c3.der\_2",

"URL=http://pki.goog/repo/certs/gts1c3.der",

"Resource=1",

"RecContentType=application/pkix-cert",

"Referer=",

"Snapshot=t7.inf",

LAST);

web\_url("gtsr1.der\_2",

"URL=http://pki.goog/repo/certs/gtsr1.der",

"Resource=1",

"RecContentType=application/pkix-cert",

"Referer=",

"Snapshot=t8.inf",

LAST);

web\_url("RapidSSLTLSRSACAG1.crt",

"URL=http://cacerts.rapidssl.com/RapidSSLTLSRSACAG1.crt",

"Resource=1",

"RecContentType=application/pkix-cert",

"Referer=",

"Snapshot=t9.inf",

LAST);

return 0;

}

**Q-9 Create a normal script of above website with correlate using hp default website.**

**ANS..** Action()

{

web\_url("index.htm",

"URL=http://127.0.0.1:1080/WebTours/index.htm",

"Resource=0",

"Referer=",

"Snapshot=t1.inf",

"Mode=HTML",

LAST);

web\_url("header.html",

"URL=http://127.0.0.1:1080/WebTours/header.html",

"Resource=0",

"Referer=http://127.0.0.1:1080/WebTours/index.htm",

"Snapshot=t2.inf",

"Mode=HTML",

LAST);

web\_url("welcome.pl",

"URL=http://127.0.0.1:1080/cgi-bin/welcome.pl?signOff=true",

"Resource=0",

"RecContentType=text/html",

"Referer=http://127.0.0.1:1080/WebTours/index.htm",

"Snapshot=t3.inf",

"Mode=HTML",

EXTRARES,

"Url=http://pki.goog/repo/certs/gts1c3.der", "Referer=", ENDITEM,

"Url=http://pki.goog/repo/certs/gtsr1.der", "Referer=", ENDITEM,

LAST);

lr\_save\_string(lr\_decrypt("6620c77998a0b4f6"), "PasswordParameter");

web\_submit\_data("login.pl",

"Action=http://127.0.0.1:1080/cgi-bin/login.pl",

"Method=POST",

"RecContentType=text/html",

"Referer=http://127.0.0.1:1080/cgi-bin/nav.pl?in=home",

"Snapshot=t4.inf",

"Mode=HTML",

ITEMDATA,

"Name=userSession", "Value=138794.996921831HVDQczHpzftVzzzHtciDfpzAfQcf", ENDITEM,

"Name=username", "Value=jojo", ENDITEM,

"Name=password", "Value={PasswordParameter}", ENDITEM,

"Name=login.x", "Value=47", ENDITEM,

"Name=login.y", "Value=8", ENDITEM,

"Name=JSFormSubmit", "Value=off", ENDITEM,

LAST);

return 0;

}

**Q-10 What is Automation Testing?**

**ANS..** Automation testing is a software testing technique where automated tools and scripts are used to execute pre-defined tests on software applications. Instead of manually executing test cases, automation testing involves the use of software to perform repetitive

and predictable tasks, such as regression testing, functional testing, performance testing, and more.

The primary goal of automation testing is to increase the efficiency and effectiveness of the testing process while reducing human intervention. Some key benefits of automation testing include:

Faster execution: Automated tests can be run much faster than manual tests, allowing for quicker feedback on the quality of the software.

Reusability: Test scripts can be reused across different builds and versions of the software, saving time and effort in test development.

Increased test coverage: Automation testing can cover a wider range of scenarios and configurations than manual testing, leading to more comprehensive test coverage.

Consistency: Automated tests perform the same steps and checks consistently, reducing the risk of human error and ensuring consistent test results.

Cost-effectiveness: While initial setup and maintenance of automated tests may require investment, in the long run, automation can save costs by reducing the need for manual testing resources.

Automation testing can be implemented using various tools and frameworks, such as Selenium for web applications, Appium for mobile applications, JUnit and TestNG for unit testing, and tools like Jenkins for continuous integration and deployment.

**Q-11 Which Are The Browsers Supported By Selenium Ide?**

**ANS..** As of my last update in January 2022, Selenium IDE, which is primarily a browser extension for Chrome and Firefox, supports the following browsers:

Google Chrome: Selenium IDE is available as a Chrome extension and can be installed from the Chrome Web Store.

Mozilla Firefox: Selenium IDE is available as a Firefox add-on and can be installed from the Mozilla Add-ons website.

These are the primary browsers supported by Selenium IDE. However, Selenium IDE may support additional browsers in the future as the tool evolves, so it's always a good idea to check the official documentation or release notes for the latest information. Additionally, Selenium WebDriver, the core automation tool ofSelenium, supports a wider range of browsers beyond just Chrome and Firefox.

**Q-12 What are the benefits of Automation Testing?**

**ANS..** Automation testing offers numerous benefits, which contribute to improving the efficiency, accuracy, and effectiveness of software testing processes. Some of the key benefits :

Time Efficiency: Automation testing can execute test cases much faster than manual testing, saving significant time in the testing process. Automated tests can run concurrently on different machines, reducing overall test execution time.

Cost Efficiency: While there is an initial investment in setting up automation frameworks and scripts, in the long run, automation testing can significantly reduce testing costs. It lowers the need for manual testers, decreases the time required for testing cycles, and minimizes the chances of defects slipping into production.

Reusability and Maintainability: Automated test scripts can be reused across different versions of the software, saving time on script creation. Additionally, when there are changes in the application, updating automation scripts is usually easier and faster compared to manual test cases, thus enhancing maintainability.

Accuracy and Consistency: Automation testing eliminates human errors inherent in manual testing, leading to more accurate and consistent test results. Automated tests perform the same steps precisely each time they run, reducing the risk of overlooking bugs due to human oversight.

Increased Test Coverage: Automation testing allows for broader test coverage by executing a large number of tests across different environments, configurations, and datasets. This helps in identifying more defects and ensuring the software's quality and reliability.

Early Bug Detection: Automated tests can be integrated into the Continuous Integration/Continuous Deployment (CI/CD) pipeline, enabling the early detection of bugs as soon as new code is added to the repository. This facilitates faster bug fixing and prevents the accumulation of bugs until later stages of development.

Supports Regression Testing: Automation testing is particularly useful for regression testing, where previously executed test cases are rerun

to ensure that new changes have not adversely affected the existing functionality. Automated regression tests can be executed quickly and frequently, providing confidence in the software's stability after each code change.

Scalability: Automation testing scales easily to accommodate growing testing needs, such as increased feature complexity or expanding test suites. It allows teams to handle larger projects without significantly increasing testing resources.

Facilitates Continuous Testing: Automation testing aligns well with the principles of Continuous Testing, which involves testing throughout the software development lifecycle. It enables rapid feedback on code changes, enhances collaboration between development and testing teams, and promotes a culture of quality across the organization.

Overall, automation testing helps organizations deliver high-quality software products more efficiently, reducing time-to-market and enhancing customer satisfaction.

**Q-13 What are the advantages of Selenium?**

**ANS..** Selenium is a popular automation testing framework primarily used for web application testing. Some of the key advantages of Selenium include:

Cross-Browser Compatibility: Selenium supports various web browsers such as Chrome, Firefox, Safari, Edge, etc.This allows testers to ensure that their web applications function correctly across different browsers.

Language Support: Selenium supports multiple programming languages including Java, Python, C#, Ruby, and JavaScript. This flexibility enables testers to write automated tests in the language they are most comfortable with.

Platform Independence: Selenium supports different operating systems like Windows, macOS, and Linux. Test scripts written on one platform can be executed on other platforms without any modifications.

Open Source: Selenium is an open-source tool, meaning it is freely available for use and can be modified according to the specific requirements of the testing project. This helps in reducing the overall testing costs.

Large Community Support: Selenium has a vast community of users, developers, and contributors who actively contribute to its development, share knowledge, and provide support through forums, blogs, and online communities. This ensures that users have access to a wealth of resources and help when needed.

Integration with Testing Frameworks: Selenium can be easily integrated with various testing frameworks like TestNG, JUnit, and NUnit. This allows for better test management, reporting, and execution.

Support for Parallel Testing: Selenium Grid allows testers to execute tests in parallel across multiple browsers and environments, significantly reducing the overall test execution time.

Rich Set of Features: Selenium offers a rich set of features for automating web applications, including interactions with web elements, handling alerts, managing cookies, and executing JavaScript code.

Continuous Integration/Continuous Deployment (CI/CD) Integration: Selenium can be seamlessly integrated with CI/CD pipelines using tools like Jenkins, Bamboo, or Travis CI, enabling automated testing as part of the development workflow.

Flexibility and Extensibility: Selenium's architecture is designed to be flexible and extensible, allowing users to extend its capabilities through custom libraries and frameworks.

**Q-14 Why testers should opt for Selenium and not QTP?**

**ANS..** Choosing between Selenium and QTP (now known as UFT - Unified Functional Testing) depends on various factors, including the project requirements, budget, skillset of the testing team, and the specific needs of the organization. Here are some reasons why testers might opt for Selenium over QTP/UFT:

• Open Source vs. Commercial Tool: Selenium is an open-source tool, meaning it's freely available for use, modification, and distribution. On the other hand, QTP/UFT is a commercial tool, which requires purchasing licenses, leading to higher upfront costs and ongoing maintenance fees.

• Platform Independence: Selenium is platform- independent and supports multiple operating systems, browsers, and programming languages. It can be used on Windows, macOS, Linux, and Unix systems, and it supports various browsers like Chrome, Firefox, Edge, Safari, etc. QTP/UFT, while versatile, may have limitations in terms of platform support and may require additional configurations or plugins for cross-platform testing.

• Programming Language Support: Selenium supports multiple programming languages, including Java, Python, C#, Ruby, and JavaScript. Testers can write test scripts in their preferred language, leveraging their existing skills and knowledge. QTP/UFT primarily uses VBScript, which may not be as widely used or preferred by all testers.

• Community Support and Flexibility: Selenium has a large and active community of developers and testers who contribute to its development, share knowledge, and provide support. This community-driven approach ensures timely updates, bug fixes, and enhancements to the Selenium framework. QTP/UFT, while supported by its vendor, may have fewer resources and community contributions.

• Integration with Testing Frameworks: Selenium can be easily integrated with popular testing frameworks such as TestNG, JUnit, NUnit, and Pytest. These frameworks provide advanced testing features such as test reporting, parallel execution, data-driven testing, and more. QTP/UFT also supports integration with testing frameworks, but Selenium's flexibility and extensibility

may offer more options and customization capabilities.

who contribute to its development, share knowledge, and provide support. This community-driven approach ensures timely updates, bug fixes, and enhancements to the Selenium framework. QTP/UFT, while supported by its vendor, may have fewer resources and community contributions**.**

• Integration with Testing Frameworks: Selenium can be easily integrated with popular testing frameworks such as TestNG, JUnit, NUnit, and Pytest. These frameworks provide advanced testing features such as test reporting, parallel execution, data-driven testing, and more. QTP/UFT also supports integration with testing frameworks, but Selenium's flexibility and extensibility may offer more options and customization capabilities.

• Web Application Testing Focus: Selenium is primarily designed for web application testing and excels in automating web browser interactions. It provides robust features for simulating user interactions, validating page content, handling alerts and pop-ups, and executing JavaScript actions. QTP/UFT, whilecapable of testing web applications, also supports testing of desktop, mobile, and API-based applications, which may be unnecessary for organizations focused solely on web application testing.

• Cost Considerations: Selenium's open-source nature eliminates licensing costs, making it a cost-effective option for organizations with budget constraints. QTP/UFT, being a commercial tool, requires purchasing licenses for each user, which can be expensive, especially for larger testing teams. Ultimately, the choice between Selenium and QTP/UFT

depends on the specific needs and priorities of the organization, as well as factors such as budget, expertise, and project requirements. While Selenium offers many advantages, QTP/UFT may still be a suitable option for organizations that prioritize commercial support, comprehensive testing capabilities, and integration with other Micro Focus products.